PWSP® User Manual

PentaSoft® Web Services Platform

Version 1.0.0.8
February 2007

http://ws.pentasoft.es
http://ws.test.pentasoft.es

PWSP® PentaSoft Web Services Platform is a registered product of PentaSoft Sistemas.
®2007 PentaSoft Sistemas. All rights reserved. Total or partial reproduction forbidden.
PentaSoft Sistemas is a PentaSoft Group company.

TABLE OF CONTENTS

1. Introduction 2

2. Technological environment identification 3

2.1. Technical features 3

3. PWSP® platform description 5

3.1. Overview 5
3.2. Available resources 7
3.3. Xml schema data (XSD file) 8
3.4. Web service formal description (WSDL file) 11
3.5. Web service discovery (DISCO file) 13
3.6. Development process help classes (HELPER file) 15
3.7. Web service public methods 16
3.8. Invocation protocols 23

4. Platform utilization examples 26

4.1. HTTP Querystring 27
4.2. Microsoft.NET Platform 32

5. Standards and rules specification 33

®2007 PentaSoft Sistemas. All rights reserved.

1. Introduction

PWSP® is the registered trademark of PentaSoft Web Services Platform, the web services platform developed by the Grupo PentaSoft and is the point of access to a service infrastructure which covers every aspect of data management and analysis.

Via the services mad available by this platform, our customers have a new generation of online tools to enable them to use powerful modules for information management and analysis, using protocols based on Internet standards and supplied via secure channels.

No advanced programming knowledge is necessary to use the platform, and as it complies with all the W3C standards relating to web services, it can be invoked from any technological environment with Internet access.

This feature enables developers to use all the power of the platform, without having to abandon their native programming environments, meaning that all the investment in current developments can be leveraged and reinforced by the new functionalities offered by the platform.

The clearest example is that you can, for example, use the PWSP® web services platform to manage the information on your webpage irrespective of the programming language used to develop it, the type of server or operating system on which it is housed or the type of information handled.

But this platform can also be used to manage all the information of the corporate application running locally on the PCs of the company's local area network; the only requirement is that the network have an Internet connection and the programming language in which the application has been developed is capable of using Internet resources (virtually all programming languages nowadays), irrespective of the operating system of each PC or the servers of the network.

The cost saving achieved by the drastic reduction of development times, a full auditing and monitoring system, together with the available of a wide range of new functionalities and the centralised information repository (accessible from anywhere) make the use of this platform's services one of the competitive advantages to be taken into account by any company using information systems.

®2007 PentaSoft Sistemas. All rights reserved.

2. Technological environment identification

2.1. Technical features

The Internet has without question been the catalyst which has changed the lives both of software developers and users, providing both with an environment accessible from virtually any location and an information repository of an unimaginable size until only recently.

The adoption of XML (e**X**tensible **M**arkup **L**anguage) as a standard for information exchange between systems is also revolutionising the communication between applications, as it offers a universal data format which means information can be adapted or transformed between differing systems.

XML web services allows applications to share information and to invoke functions of other applications irrespective of how these applications were created, the operating system or the platform on which they run, and the devices used to access them. Although XML web services are independent from each other, they can be linked to create a collaborative group to carry out a given task.

Based on the above, PentaSoft has created a complete platform based on XML web services which work together to allow access to the applications of our customers to the various information repositories managed and administered by any of the group companies. We have named this platform PWSP® (**P**entaSoft **W**eb **S**ervices **P**latform).

The PWSP® platform's architecture is shown below:

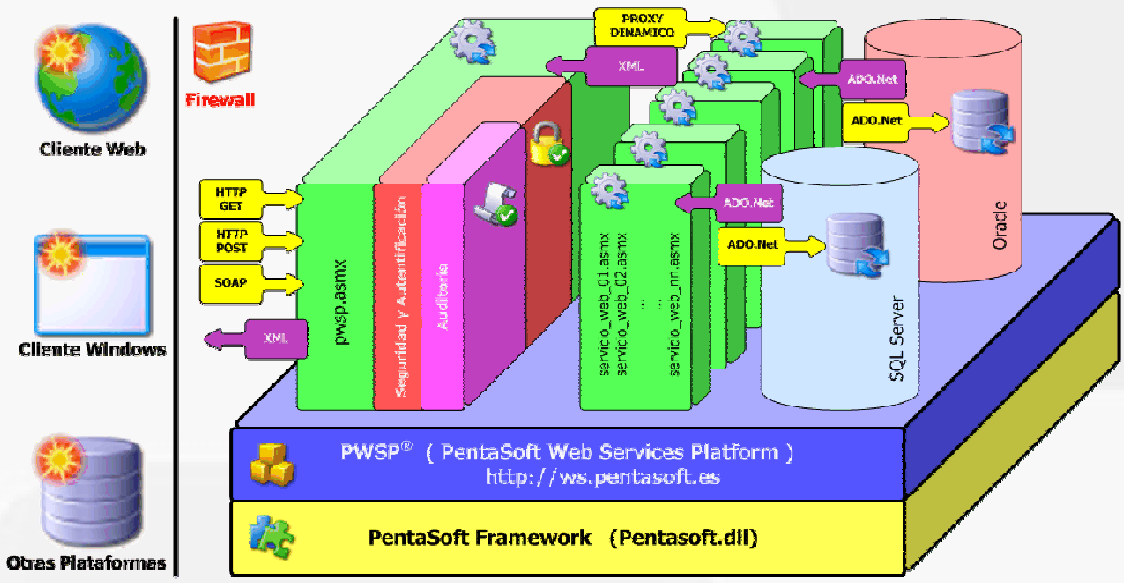


figure 1

As mentioned above, the aim of the platform is to offer access to the information available via XML web services, and from any location and technological environment.

It is accepted as a general standard that the web services are invoked on the Internet using standard XML-based protocols. In any event, in order to work with the platform, we may use two types of calls:

®2007 PentaSoft Sistemas. All rights reserved.

- Standard HTTP calls (both HTTP GET and HTTP POST) using querystrings and forms.
- Call via the SOAP protocol (versions 1.1 and 1.2) on HTTP.

The first type of call consists of HTTP requests in a manner similar to a standard Internet browser. Requests are built starting with the URL (**U**niversal **R**esource **L**ocator) of the entry point to the web service, then adding query arguments to specify the values to be supplied to the web service using the standard**?parameter=value** format.

The call return consists of a stream in XML format which must be analysed subsequently by the customer application making the call, obtaining the response to the information requested, together with the return codes and additional information. We can also use forms to send requests and subsequently reception of the information.

This way of invoking the platform's web services is the simplest method and requires no special programming or the use of a specific protocol. We shall later provide detailed examples of calls to the platform's web services using this invoking system.

The second type of call is based on classic, XML-based standards. The PWSP® platform uses **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol) as the standard access protocol in calls to services, given its completeness, independence from the transport protocol and security.

SOAP is an XML-based protocol, which allows interaction between various devices and is able of transmitting complex information. The data can be transmitted via HTTP, SMTP, etcetera and a format is specified for messages. It comprises an **envelope**, whose structure is formed by the following elements: **header** and **body** as shown in the illustration below:

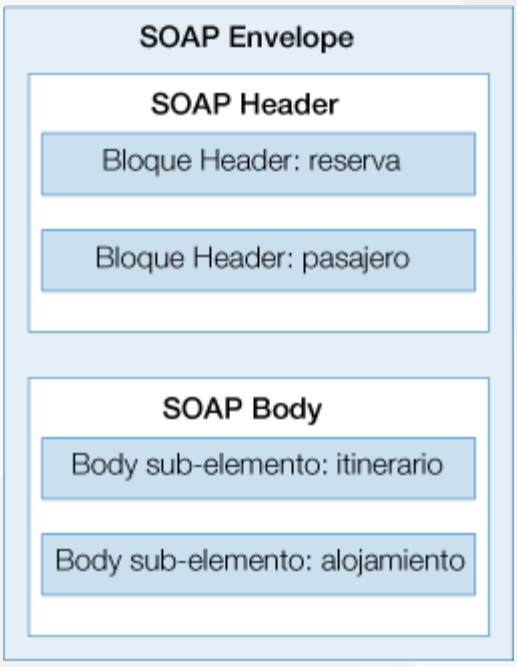


figure 2

Later, in the case study section, we shall show an example of invoking the web services of the platform using SOAP, together with an example of invoking web services via HTTP with a querystring and also using HTTP with a form.

®2007 PentaSoft Sistemas. All rights reserved.

3. PWSP® platform description

3.1. Overview

The platform consists of a single public XML web service which is the access point to same. This service gathers the information via an XML entry and, based on the information provided in the call parameters, is capable of invoking the internal services.

Continuing with the description of the platform as shown in **figure 1** calls to XML web services can be made from virtually any customer application, whether it is a web application or a desktop application running locally and has some means of connecting to the Internet.

As the access is Internet-enabled, we must know where to locate this service. Simply put, we may say that in the same way that access to a webpage is defined by an URL (**U**niform **R**esource **L**ocator) which is its address, a web service is defined by an **URI** (**U**niform **R**esource **I**dentification) and its interface, via which it can be accessed. On the PWSP® platform the URI for the platform access service is:

http://ws.pentasoft.es/pwsp/pwsp.asmx

The platform also has a test environment, similar to the production environment, for conducting the necessary test before to putting the modules into production. The test environment URI is:

http://ws.test.pentasoft.es/pwsp/pwsp.asmx

As mentioned above, the fact that access is gained via a single entry web service does not prevent us from using it to access a wide range of services specifically developed for each customer.

As we shall see, single access will afford us greater and better control of processes relating to security, auditing and extensibility, enabling us to offer new modules without having to perform modifications to the authentication and call methods, carrying the content supplied using the same methods.

When invoking the web service which defines the access point to the platform using any browser and the above URI, we can see a webpage which enables us to verify service availability, together with several of its features such as available methods, protocols supported, generic description of same, documentation on possible return codes and many others.

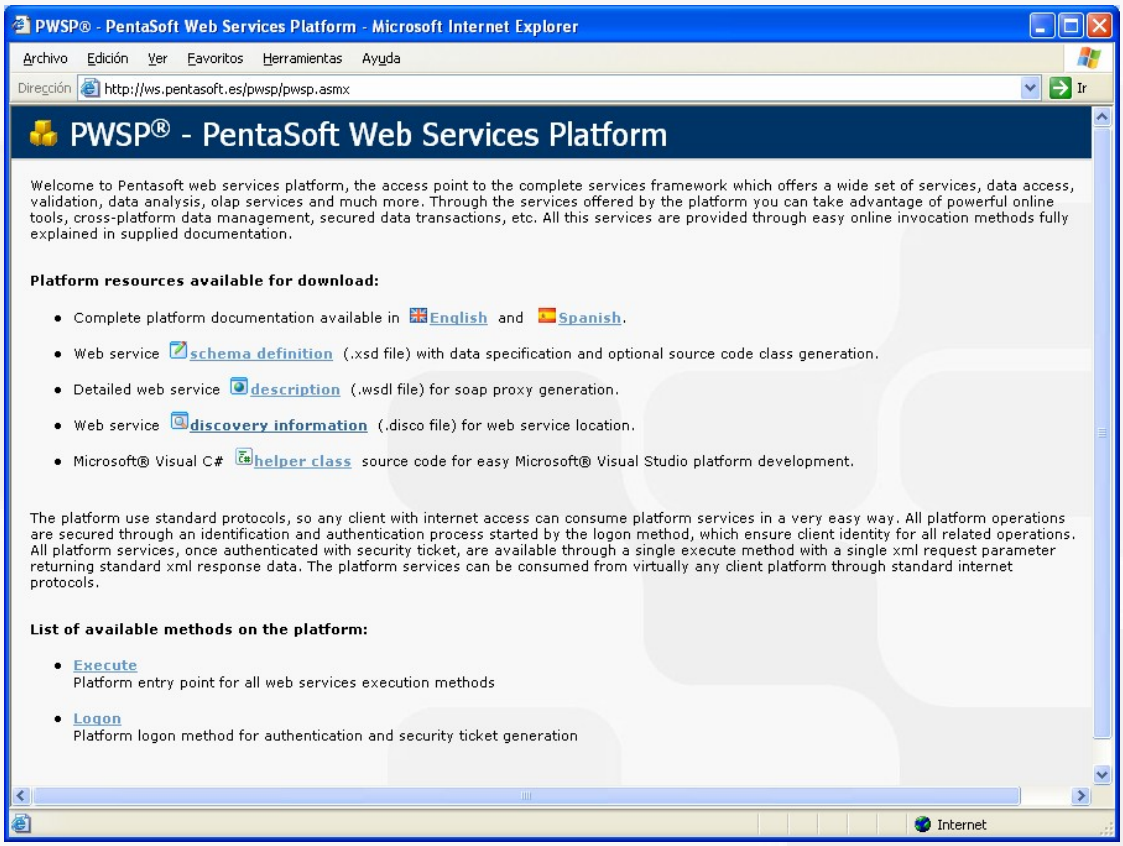
From this page we can carry out all the tests we need using any of he web services implemented on the platform, obtaining exactly the same results as when we make calls from the technological environment of each customer in a program customer, with the same information supplied on security, auditing, etcetera.

In fact, the mentioned page natively calls the platform in the same was as a customer program, by offering the results in a new browser page being used to make the calls and in the standard XML format, so as to check the value returned by the platform, without the customer writing a single line of code.

This platform feature educes customer development time, as all the specification can be performed on the expected functioning of the web service, performing all the necessary tests and checking the real data which the service will return and all the possible return codes without needing to write a single line of code, so that when the programmer starts work, he can be sure of the behaviour and the values to be supplied and received by the web service.

®2007 PentaSoft Sistemas. All rights reserved.

The platform access page is as follows:


PWSP® - PentaSoft Web Services Platform - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Dirección http://ws.pentasoft.es/pwsp/pwsp.asmx
Ir
PWSP® - PentaSoft Web Services Platform
Welcome to Pentasoft web services platform, the access point to the complete services framework which offers a wide set of services, data access, validation, data analysis, olap services and much more. Through the services offered by the platform you can take advantage of powerful online tools, cross-platform data management, secured data transactions, etc. All this services are provided through easy online invocation methods fully explained in supplied documentation.
Platform resources available for download:
Complete platform documentation available in English and Spanish.
Web service schema definition (.xsd file) with data specification and optional source code class generation.
Detailed web service description (.wsdl file) for soap proxy generation.
Web service discovery information (.disco file) for web service location.
Microsoft® Visual C# helper class source code for easy Microsoft® Visual Studio platform development.
The platform use standard protocols, so any client with internet access can consume platform services in a very easy way. All platform operations are secured through an identification and authentication process started by the logon method, which ensure client identity for all related operations. All platform services, once authenticated with security ticket, are available through a single execute method with a single xml request parameter returning standard xml response data. The platform services can be consumed from virtually any client platform through standard internet protocols.
List of available methods on the platform:
Execute
Platform entry point for all web services execution methods
Logon
Platform logon method for authentication and security ticket generation
Internet
figure 3

®2007 PentaSoft Sistemas. All rights reserved.

3.2. Available resources

As shown in figure 1 in the previous section, we can find several links: the first refer to the documentation available for the platform such as this document, initially available in both Spanish and English.

This document provides an initial description of the platform, which focuses on explaining the application's technological environment, as well as an overview of its workings and features available to customers.

This document will be enhanced in subsequent versions with additional information on alternatives when using the platform, relating to major aspects such as security, auditing, monitoring and many others, and which will enable every customer to meet the most demanding requirements in these and other areas.

®2007 PentaSoft Sistemas. All rights reserved.

3.3. Xml schema data (XSD file)

On the access page there is a link to the schema definition (.xsd file) which will supply us with the necessary information to exchange information correctly.

When we referred to XML as a standard for information exchange between dissimilar systems, at the beginning of this chapter, we stated that this exchange was based on both systems using schemas to define the data structures and types to be shared.

These schemas are defined as files (**X**ml **S**chema **D**efinition). An XML schema (XSD file) defines the structure of an XML document. The purpose of an XML schema can be summarised as:

- It defines the elements which may appear in a document
- It defines the attributes which may appear in a document
- It defines which elements are child elements of other elements
- It defines the order of the child elements
- It defines the number of child elements
- It defines whether an element is empty or can include text
- It defines the types of element and attribute data
- It defines the fixed and default values for elements and attributes

Therefore, in order to specify or read any value which we wish to supply or read from the web service, we need to consult the structure and type of data the web service will accept and return, and which in the case of the platform has the following contents:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Logon" type="Logon" />
<xs:complexType name="Logon">
	<xs:sequence>
		<xs:element minOccurs="0" maxOccurs="1" name="Culture" type="PlatformCulture" />
		<xs:element minOccurs="1" maxOccurs="1" name="Login" type="xs:string" />
		<xs:element minOccurs="1" maxOccurs="1" name="Password" type="xs:string" />
	</xs:sequence>
</xs:complexType>
<xs:element name="Request" type="Request" />
<xs:complexType name="Request">
	<xs:sequence>
		<xs:element minOccurs="1" maxOccurs="1" name="Ticket" type="xs:string" />
		<xs:element minOccurs="0" maxOccurs="1" name="Culture" type="PlatformCulture" />
		<xs:element minOccurs="1" maxOccurs="1" name="WebServiceId" type="xs:string" />
		<xs:element minOccurs="1" maxOccurs="1" name="MethodId" type="xs:string" />
		<xs:element minOccurs="1" maxOccurs="1" name="Data" type="xs:string" />
	</xs:sequence>
</xs:complexType>
<xs:simpleType name="PlatformCulture">
	<xs:restriction base="xs:string">
		<xs:enumeration value="Neutral" />
		<xs:enumeration value="en_US" />
		<xs:enumeration value="es_ES" />
	</xs:restriction>
</xs:simpleType>
<xs:element name="Response" type="Response" />
<xs:complexType name="Response">
	<xs:sequence>
		<xs:element minOccurs="1" maxOccurs="1" name="WebServiceId" type="xs:string" />
		<xs:element minOccurs="1" maxOccurs="1" name="MethodId" type="xs:string" />
```


®2007 PentaSoft Sistemas. All rights reserved.

```
                <xs:element minOccurs="1" maxOccurs="1" name="Code" type="xs:string" />
                <xs:element minOccurs="1" maxOccurs="1" name="Description" type="xs:string" />
                <xs:element minOccurs="1" maxOccurs="1" name="Message" type="xs:string" />
                <xs:element minOccurs="0" maxOccurs="1" name="Data" type="xs:string" />
        </xs:sequence>
</xs:complexType>
<xs:element name="ResponseSet" type="ResponseSet" />
<xs:complexType name="ResponseSet">
        <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="1" name="Version" type="xs:string" />
                <xs:element minOccurs="1" maxOccurs="1" name="ResponseItems" type="ArrayOfResponseItem" />
        </xs:sequence>
</xs:complexType>
<xs:complexType name="ArrayOfResponseItem">
        <xs:sequence>
                <xs:element minOccurs="0" maxOccurs="unbounded" name="ResponseItem" type="ResponseItem" />
        </xs:sequence>
</xs:complexType>
<xs:complexType name="ResponseItem">
        <xs:sequence>
                <xs:element minOccurs="1" maxOccurs="1" name="Key" type="xs:string" />
                <xs:element minOccurs="1" maxOccurs="1" name="Value" type="xs:string" />
        </xs:sequence>
</xs:complexType>
<xs:element name="ResponseItem" type="ResponseItem" />
</xs:schema>
```

The file defines five fundamental elements which define the way in which we need to supply and gather the information we exchange via the web service:

- **Logon**: This defines the element for supplying information to access the platform. It will return a security ticket to be supplied with the following requests so as to access the platform services.
- **Request**: This defines the element for supplying information specified by the type of request and the data we are going to send to the web service. It also defines the internal web service to be executed, the necessary data and the security ticket and language in which we wish to communicate with the web service.
- **Response**: This defines the element which will contain the response returned by the web service as a result of the calls we have made. It will inform us via codes and messages of the information requested and the response status.
- **ResponseSet**: This defines the set of possible replies when the request made produces a result containing multiple elements.
- **ResponseItem**: This is each of the elements within a multiple value response.

The file also defines certain simple types which specify the subset of data we can supply in the parameters of the elements defined:

- **PlatformCulture**: This defines the various languages supported for communications with the platform's web services.

The advantage of the platform is that this data schema will not change, independently of internal web service invoked, so that the functionality can be enlarged without any type of restriction or need to change the manner in which we call the platform, only the values contained in the appropriate **Request** element.

®2007 PentaSoft Sistemas. All rights reserved.

For example, the entry process into the system is carried out by invoking the **Logon** method of the web service, to which we need to send a Logon element as additional information, specifying the **Login** and **Password**, together with the language in which we wish to communicate with the platform.

In response, we will receive a **Response** element with the information on whether the call was a success; if so, the security ticket within the **Data** element contained in the **Response** element.

After obtaining the security ticket we can start to make calls to the platform's web services by invoking the **Execute** method of the web service, to which we will send the **Request** element as information.

As can be seen from the example, the definition of the data layout (XSD files) is fundamental when exchanging information with the platform and in general, with any system based on calls to XML web services.

®2007 PentaSoft Sistemas. All rights reserved.

3.4. Web service formal description (WSDL file)

The web service formal definition is to be found in the link marked **web service description** in which it shows the protocols supported for making the call to the service. The information appears in the following form:

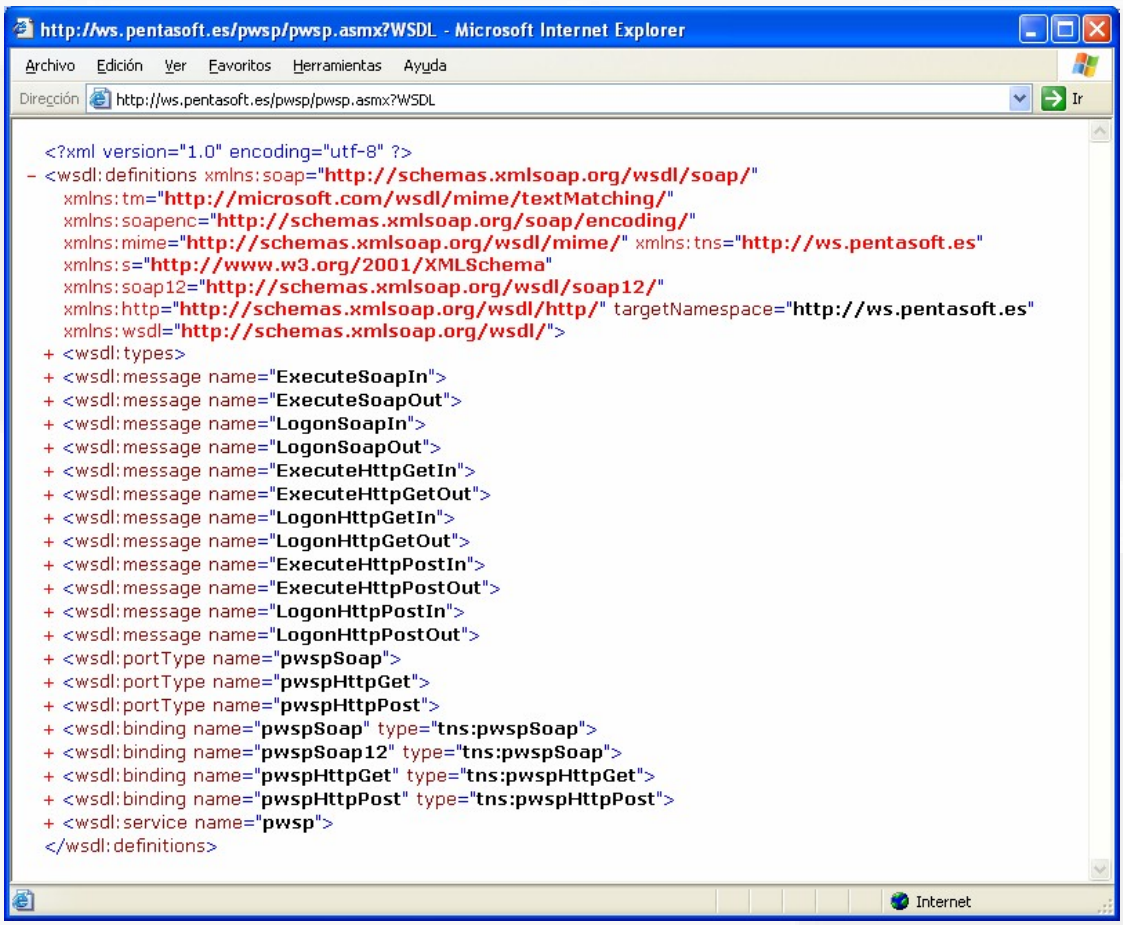


figure 4

This is the formal specification in XML format of the WSDL document (**W**eb **S**ervice **D**escription **L**anguage).

We should remember that this information is only necessary for calls made using the **SOAP** protocol, as in calls made using querystrings or forms over HTTP it is of no use, as the invocation and response are made using standard HTTP calls.

In order to explain the contents of this specification of the WSDL document and its utility, we must start by defining what a web service description language is.

This service description language makes it possible to take use the advantages of SOAP, by providing a simple means of collaboration between web service providers and the users of same.

®2007 PentaSoft Sistemas. All rights reserved.

Bearing in mind that SOAP is a standard for the communication of pure contents, it needs a language to describe this content, which is where WSDL comes into its own as a description language.

SOAP messages carry information on the type, so SOAP takes into account the dynamic establishment of same. However, a function cannot be called up correctly unless the name, number of parameters and type of each one are known.

This is really the WSDL main function, to establish a contract between customer and server, which both know and strictly respect in the information exchange.

With WSDL we can therefore automate the generation of proxy servers for web services in a manner totally independent of the language and the platform.

We must bear in mind that while WSDL has been designed to be able to express links to protocols other than SOAP, our main concern in this case is WSDL as regards its relation with SOAP via HTTP.

In addition, while SOAP is currently used mainly for calls to functions or remote procedures, WSDL allows the specification of documents for transmission via SOAP.

Below we shall illustrate the structure of WSDL, an XML document, showing the relationships between the five sections which form the WSDL document.

This document can be divided into two groups of sections. The upper group comprises the abstract definitions, while the lower group comprises specific descriptions. The abstract sections define the SOAP messages independently of the platform and language; they do not contain specific elements of the language or the equipment. This facilitates the definition of a set of services which can be implemented by several different websites.

The detailed analysis of the WSDL document of the web service to access the PWSP® platform falls outside the scope of this document, though several examples of their content and use are shown in the examples section.

As a repository of resources, several URL are given with information about the web services and protocols for implementation of same:

- Specification WSDL 1.1
- Specification SOAP 1.1
- Specification SOAP 1.2

®2007 PentaSoft Sistemas. All rights reserved.

3.5. Web service discovery (DISCO file)

After having seen the formal definition of the Web service, potential users must be able of discovering where the Web service is located and how it operates.

For this purpose, we can use DISCO, a Microsoft® technology for publishing and discovering Web services.

DISCO defines the format of an XML document and an interrogation algorithm of same, making it possible to discover the Web services published on a given server. DISCO also makes it possible to discover the capabilities of each of the Web services via the documentation and the way in which it interacts with same via its WSDL specification.

It is not compulsory to use DISCO; it is offered as an alternative to discover the Web service and establish a reference to the platform. It is another option to help developers working with Microsoft® Visual Studio, as these development environment includes tools which use the information contained in this file to obtain data on the Web service, such as its location, features, etcetera.

The DISCO file contents for the platform Web service is as follows:

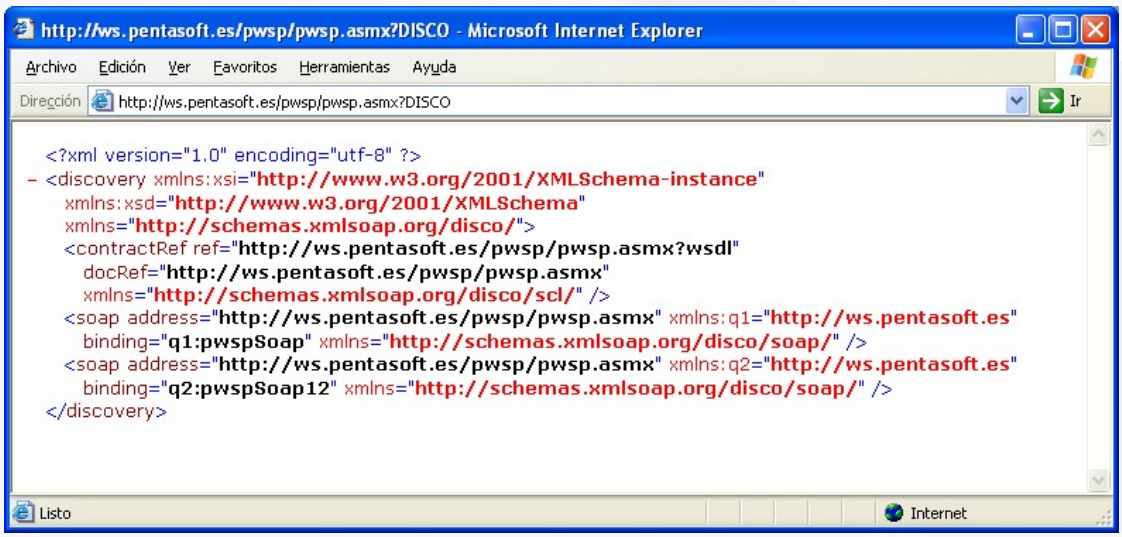


figure 5

We currently have two utilities within Microsoft® Visual Studio to take advantage of the information contained in the DISCO file. One is the command line utility called disco.exe and the other is the Add Web Reference from the programming environment of the tool.

In both cases the objective is to achieve the necessary information to correctly locate the Web service, and obtain the functionalities of same from its WSDL description file.

Once we have information on the methods implemented in the web service, and the documentation on its parameters and type, we can define what we know as the Proxy.

®2007 PentaSoft Sistemas. All rights reserved.

This class will enable us to make calls to the Web service methods, and collect the values which the calls return, which is the basis for establishing communication with the Web service.

As mentioned above, the information contained in the DISCO file is a utility which is basically for developers who work with Microsoft® Visual Studio, so we shall show a complete example in the section of examples with .Net technology

®2007 PentaSoft Sistemas. All rights reserved.

3.6. Development process help classes (HELPER file)

When we call the platform from Microsoft® Visual Studio environments, we have a C# source code file with classes definition and the necessary codes to perform operations with the platform, handling data and other tasks.

To do so we need to incorporate this file into our Microsoft® Visual Studio project, and we can start to use the classes and utilities it contains.

The information appears as follows:

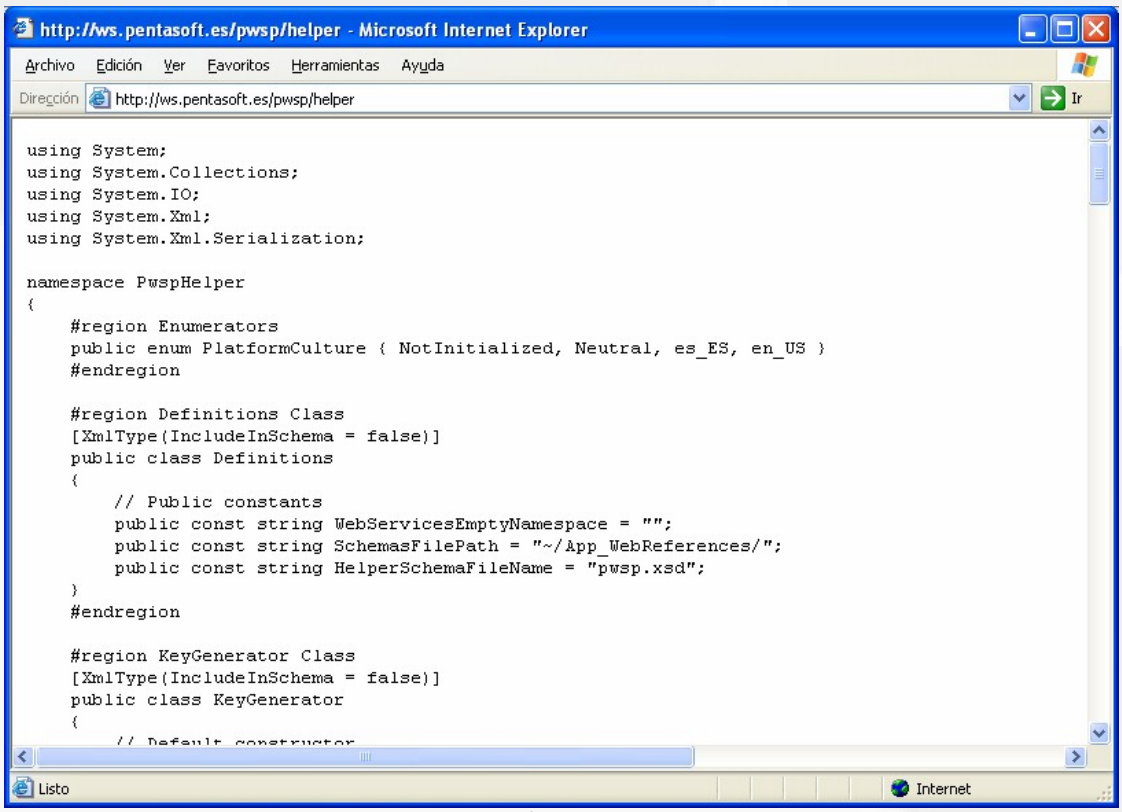


figure 6

Within these help classes, there are also classes with utilities to perform object serialisation and deserialisation operations in xml streams, which is the standard format accepted by the platform.

We should remember that this information is only necessary for calls made using the **SOAP** protocol, as in calls made using querystrings or forms over HTTP there is no utility, as the invocation and response are made using standard HTTP calls.

®2007 PentaSoft Sistemas. All rights reserved.

3.7. Web service public methods

As we saw in figure 3, the last section which appeared on the page showed a list with the methods offered by the platform:

- **Logon:** This allows access to the platform for identification, authentication and obtaining the security ticket to make the calls to the internal platform Web services.
- **Execute:** This executes the calls to the internal services of the platform, performs the information audit and management processes and returns the information compiled from the internal Web services of the platform.

When calling up the Logon method, we find the following screen:

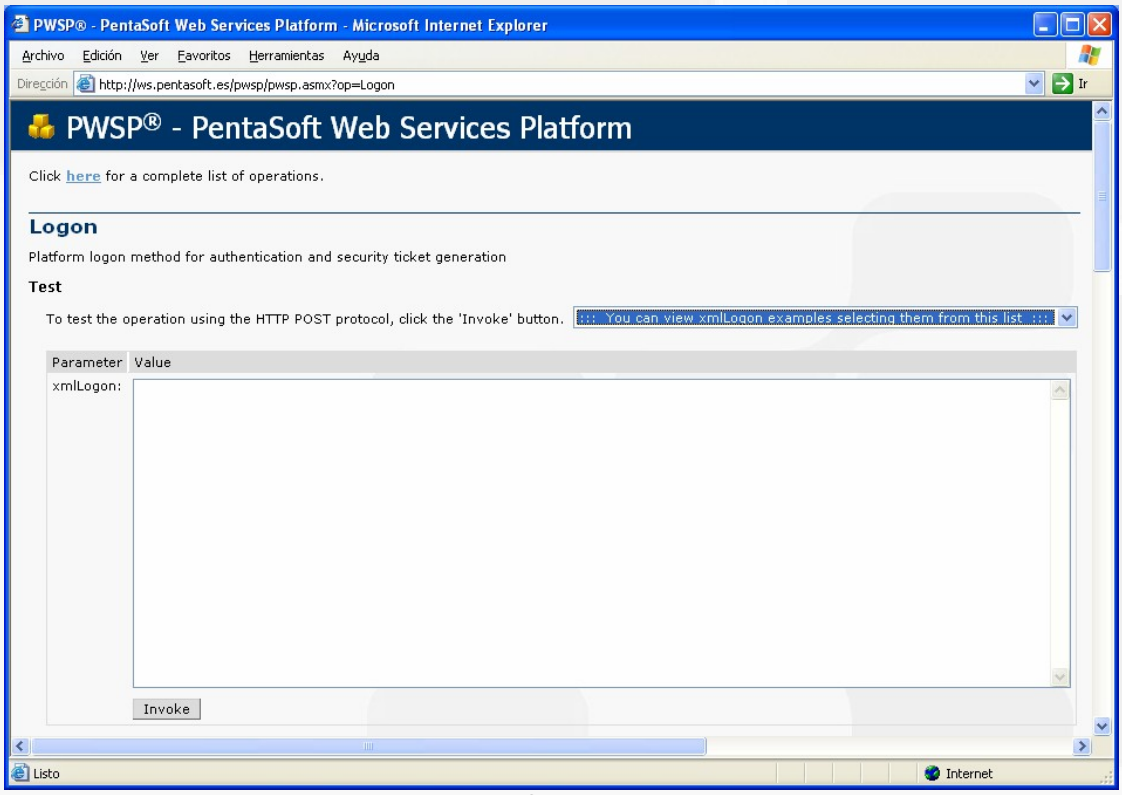

PWSP® - PentaSoft Web Services Platform - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Dirección http://ws.pentasoft.es/pwsp/pwsp.asmx?op=Logon
Ir
PWSP® - PentaSoft Web Services Platform
Click here for a complete list of operations.
Logon
Platform logon method for authentication and security ticket generation
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
::: You can view xmlLogon examples selecting them from this list :::
Parameter
Value
xmlLogon:
Invoke
Listo
Internet

figure 7

As we can see, the only information shown is the description of the method itself, together with a dropdown list where we can use an example of those offered to see the appearance of the information we must enter in the **xmlLogon** parameter which is just the result of the serialisation as an XML stream text of the Logon element specified in the schema file (XSD).

To check the appearance, we select any example from the list and the parameter is automatically filled in with the data of the example with a demo user which the platform supplies.

The screen then shows the xmlLogon parameter filled in with the information referring to the demo user, which consists of his Login and Password, as well as the language in which we are going to communicate with the platform.

®2007 PentaSoft Sistemas. All rights reserved.

The screen appears as follows:

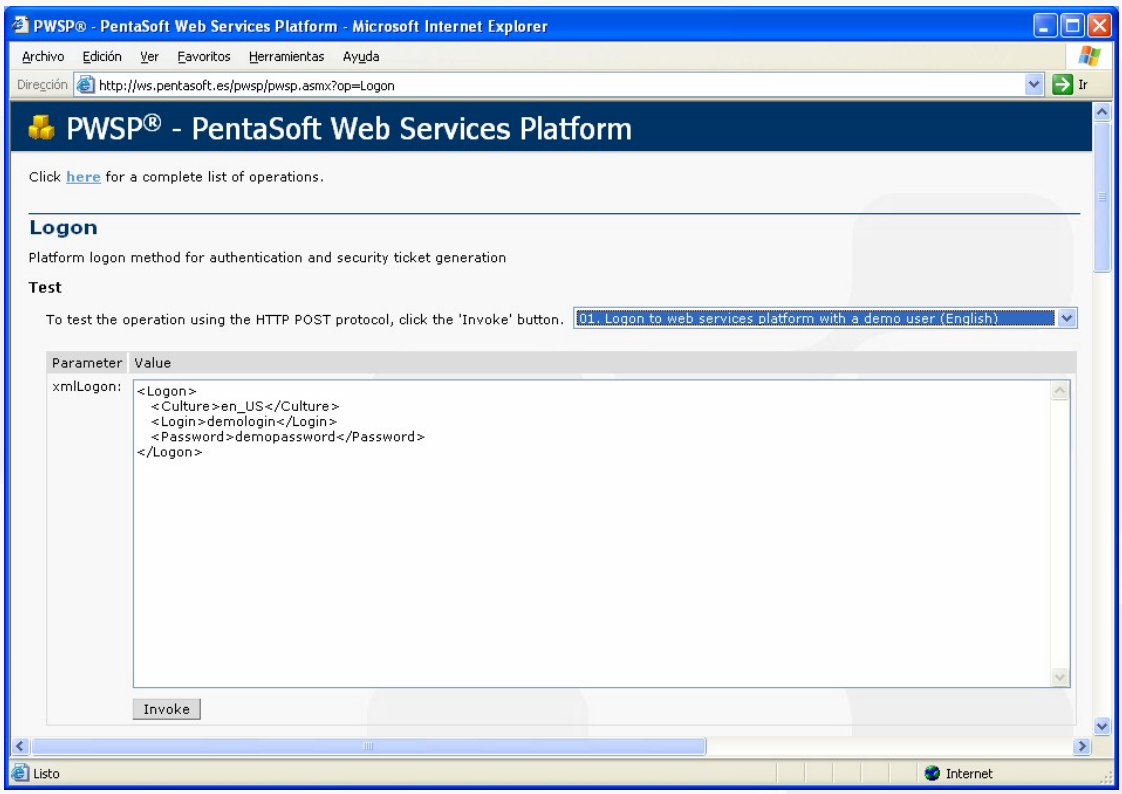


figure 8

As we can see, the text stream is just the serialisation of a **Logon** type of object as specified in the schema definition file (XSD) which we saw in section 3.3.

We can, of course, change any value of the contents of the parameter to carry out the necessary tests, entering our own values for the Login and Password, after PentaSoft has supplied us with our own user and password to use the platform.

After entering the required data, we can call the platform web service. To do so, we only need press the **Invoke** button which will invoke the internal identification, authentication service and generation of the security ticket.

This invocation is made using the HTTP POST protocol to facilitate the test process, but any of the three protocols supported could be used, such as HTTP GET, SOAP 1.1 or SOAP 1.2.

If we have correctly specified the values of the parameter and have authorisation to access the platform, we will see a new window in our browser which will inform us of the access process and provide us with the necessary security ticket to make the following calls for execution of Web services on the platform. The screen will appear as follows:

®2007 PentaSoft Sistemas. All rights reserved.

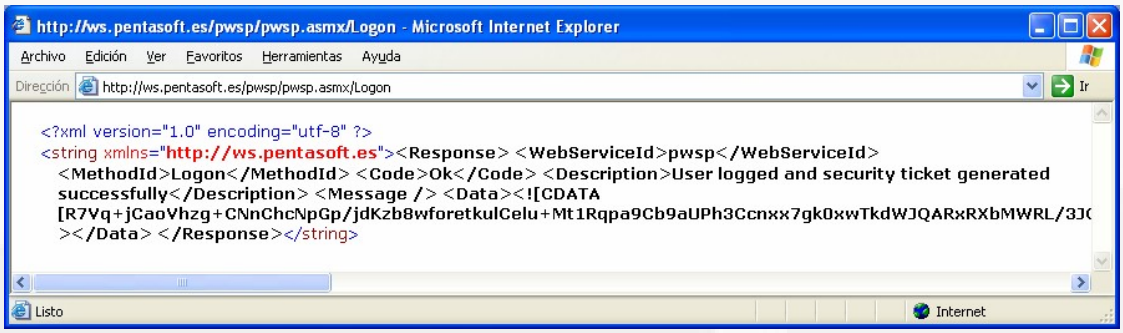


figure 9

As we can see, the response is another XML document serialised as a text stream and which contains a **Response** element in accordance with the specified schema file (XSD) and which in turn contains the following elements:

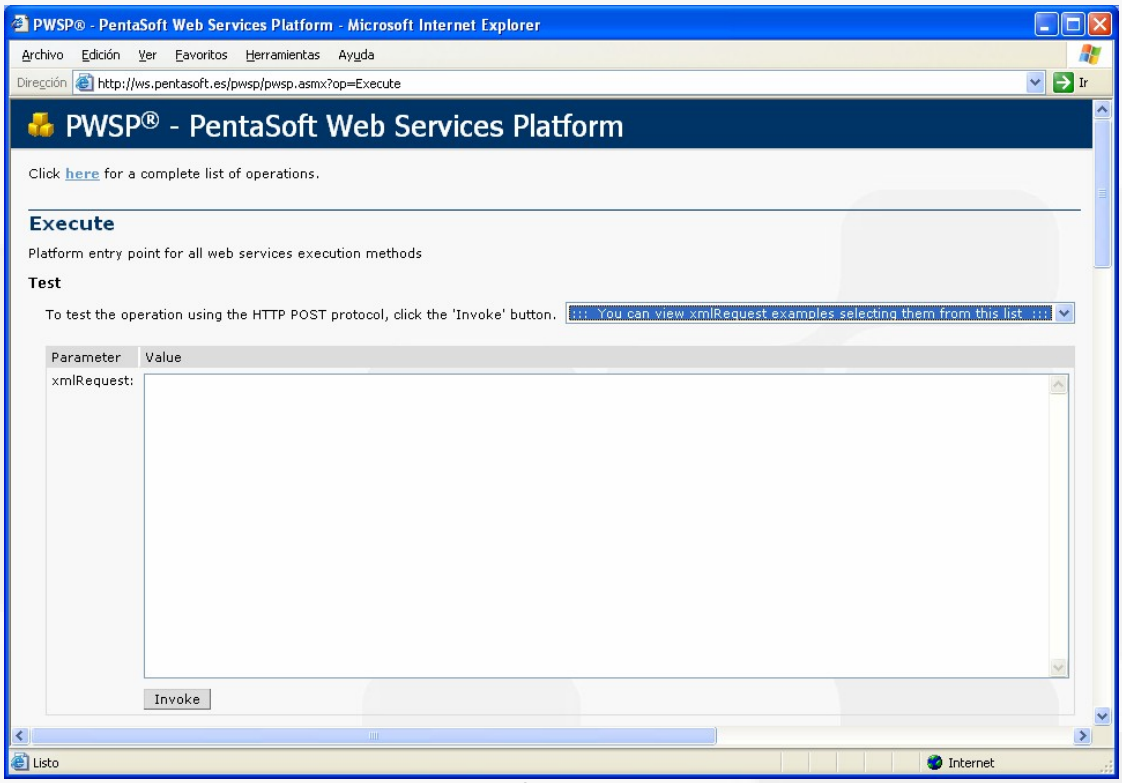
- **WebServiceId**: Web service identifier which made the call.
- **MethodId**: The method invoked within the mentioned web service.
- **Code**: The call return code, which is marked OK in this case as it was correct.
- **Description**: Call result description, which in this case reports that the user access was correct and the security ticket was correctly generated.
- **Data**: Contains call response information, and in the case of the Logon method, will contain the value of the security ticket, which is an encrypted stream of characters which we must copy exactly if we wish to make the calls to the internal Web services methods of the platform. The ticket value is specified between special identifiers in the following manner <![CDATA[**R7Vq+.....ZJel**]]> where the value of the ticket corresponds **solely** to the characters highlighted in red, excluding the characters <![CDATA[at the beginning of the text and]]> end of same. It is important to select only the specified characters, as the inclusion or exclusion of any of them will cause the following calls to fail, as no valid ticket will be found to execute them.

Once we have carried out the platform access process and obtained our security ticket, we can execute methods of the internal Web services offered by the PWSP® platform using the other public method offered.

In order to access this method we simply click on the **Execute** link to access the page where we can enter the information to invoke the internal Web services of the platform.

When clicking on this link, the following screen appears:

®2007 PentaSoft Sistemas. All rights reserved.


PWSP® - PentaSoft Web Services Platform - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Dirección http://ws.pentasoft.es/pwsp/pwsp.asmx?op=Execute
Ir
PWSP® - PentaSoft Web Services Platform
Click here for a complete list of operations.
Execute
Platform entry point for all web services execution methods
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
::: You can view xmlRequest examples selecting them from this list :::
Parameter
Value
xmlRequest:
Invoke
Listo
Internet
figure 10

As we can see, the appearance is very similar to the platform access page, with the difference in this case that the name of the parameter requests information on the information to be included in the request.

We also have a list with several examples available:

Pentasoft development Framework examples:

- **01. Framework (Pentasoft.dll) GetFrameworkResponseSet (English)**: This returns the set of possible values of return codes which the development framework can generate, with the descriptions of same in English.
- **02. Framework (Pentasoft.dll) GetFrameworkResponseSet (Spanish)**: This returns the set of possible values of return codes which the development framework can generate, with the descriptions of same in Spanish.
-

PWSP platform® examples:

- **10. Platform (pwsp.xsd) GetPlatformSchemaDefinition**: This returns an XML stream with the file information which defines the XSD schema for using the platform.
- **11. Platform (pwsp.asmx) GetPlatformMethods**: This returns an XML chain with the information on all the methods which can be invoked on the platform.
- **12. Platform (pwsp.asmx) GetPlatformResponseSet (English)**: This returns a ResponseSet element serialised as an XML text stream with the information on all the possible values of the return codes which the platform can return with the descriptions of same in English.

®2007 PentaSoft Sistemas. All rights reserved.

- **13. Platform (pwsp.asmx) GetPlatformResponseSet (Spanish)**: This returns a ResponseSet element serialised as an XML text stream with the information on all the possible values of the return codes which the platform can return with the descriptions of same in Spanish.
- .
- **14. Platform (pwsp.asmx) GetPlatformXmlDocumentation**: This returns an XML text stream with the technical documentation on the platform modules and components.
- **15. Platform (pwsp.asmx) GetPlatformHelperClasses**: This returns, within the Data element of the Response object, the C# code with the classes and necessary code to call the platform from Microsoft® Visual Studio environments.
- **16. Platform (pwsp.asmx) GetPlatformTestData**: This returns, within the Data element of the Response object, a text file with the xml structures of the example to call the platform from any environment.
- **17. Platform (pwsp.asmx) GetPlatformVersions**: This returns an XML text chain with the documentation on the different versions of the platform and the modifications and upgrades made to same.

Pstsis06009 demo web service examples:

The PWSP® platform presents demo web service called pstsis06009 which carries out the basic management of a customer with methods for adding, eliminating or querying a customer which only has a name, surname and age. This demo web service is fully functional and performs real operations on the demo database, so it is a real example of the functioning of the platform.

- **20. Demo (pstsis06009.xsd) GetWebServiceSchemaDefinition**: This returns an XML stream with the file information which defines the XSD schema for using the demo web service. This method will be implemented on all the Web services developed on the platform.
- **21. Demo (pstsis06009.asmx) GetWebServiceMethods**: This returns an XML stream with the information on all the method which can be invoked within the demo web service. This method will be implemented on all the Web services developed on the platform.
- **22. Demo (pstsis06009.asmx) GetWebServiceResponseSet (English)**: This returns a ResponseSet element serialised as an XML text stream, with the information on all the possible return codes which the demo web service can return and with the descriptions of same in English. This method will be implemented on all the Web services developed on the platform.
- **23. Demo (pstsis06009.asmx) GetWebServiceResponseSet (Spanish** This returns a ResponseSet element serialised as an XML text stream, with the information on all the possible return codes which the demo web service can return and with the descriptions of same in Spanish. This method will be implemented on all the Web services developed on the platform.
- **24. Demo (pstsis06009.asmx) GetWebServiceXmlDocumentation**: This returns an XML text stream with the technical documentation concerning the methods, constants and other components of the demo web service. This method will be implemented on all the Web services developed on the platform.
- **25. Demo (pstsis06009.asmx) GetWebServiceHelperClasses**: This returns within the Data element of the Response object the C# code with the classes and necessary code to call the web service methods from Microsoft® Visual Studio environments.
- **26. Demo (pstsis06009.asmx) GetWebServiceTestData**: This returns, within the Data element of the Response object, a text file with the sample xml structures to call the web service methods from any environment.
- **27. Demo (pstsis06009.asmx) AddCustomer**: This is used to add the data of a customer as specified in the call in the demo database.

®2007 PentaSoft Sistemas. All rights reserved.

- **28. Demo (pstsis06009.asmx) RemoveCustomer**: This is used to delete the data of the customer specified in the call to the demo database.
- **29. Demo (pstsis06009.asmx) QueryCustomer**: This is used to query the data of the customer specified in the demo database call.

In the future, more examples will be added with new aspects for calls to the platform, which can be modified with those of each customer so as to make it simpler to perform all the necessary tests before starting the implementation process of the calls on the part of the customer.

To illustrate the appearance of the screen which shows the sample of the invocation process of a web service method, we are going to select from the list of examples, option 25 to see how the functioning of the process is performed to add a customer to the demo database, with the invocation of a demo web service method pstsis06009 implemented on the platform.

The screen would appear as follows:

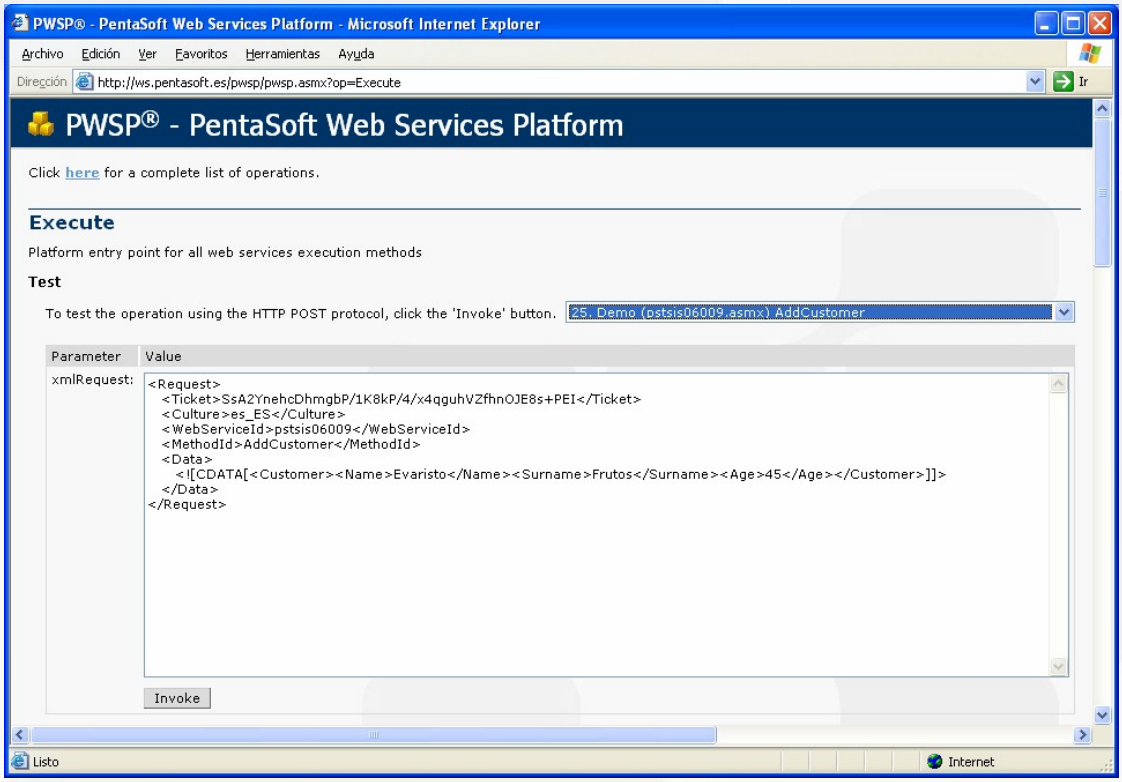


figure 11

As we can see, the text stream to be entered in the xmlRequest parameter is a serialised Request element, which inside its Data element also contains the XML specification of the Customer element, which has been defined in the XSD file for this demo web service and which specifies what a customer must have to be managed via the web service.

We can see how the Request element will always be the same as regards its schema, as defined in the platform's XSD file, and the only item to be changed in each case is the identifier of the web service and the invocation method.

®2007 PentaSoft Sistemas. All rights reserved.

The other information which will vary is that contained in the **Data** element and which will define the structure and values to invoke the web service method, and which in this case is a **Customer** element which in turn is defined in its schema (XSD file) provided by the web service pstsis06009.

We can see how the value entered in the **Data** element is enclosed between special characters **<![CDATA[.....]]>**, these characters enable us to specify information of any type, which is not interpreted by the XML parser, avoiding any misinterpretation of the XML elements defined in the call parameters.

It is important to emphasize that it is **essential** to specify these escape characters within the **Data** element of the calls to the Web services so that the platform can correctly interpret the information contained in same.

Once the xmlRequest parameter is filled in with the necessary information, we can click on the **Invoke** button to call the web service.

If we have filled in the information correctly, and after making the invocation call, we obtain a screen as follows:

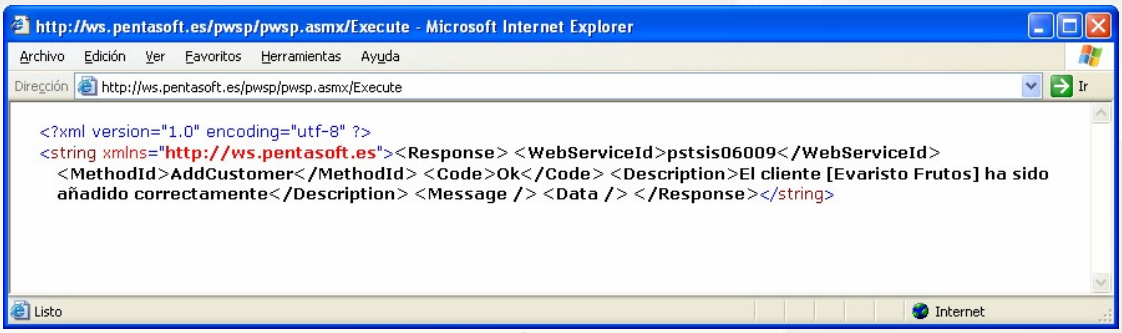


figure 12

Here we can see how we again obtain a serialised Response element with the execution return code OK and the description, informing us that the customer has been correctly added to the database.

This invocation method using the Execute method of the platform will be the one we use always to call any web service implemented in same, so that the only information we must vary is the identifier of the web service, the method to be executed and the specific information within the Data element of the request.

®2007 PentaSoft Sistemas. All rights reserved.

3.8. Invocation protocols

In the invocation examples we have seen earlier, we have used the HTTP POST protocol to call the Web services, as this is a simple protocol and does not need to build any type of special transport structure.

But to invoke the platform, Web services we have four protocols:

- HTTP GET
- HTTP POST
- SOAP 1.1
- SOAP 1.2

The choice of each one will depend on the programmer performing the call, but in all cases the level of functionality obtained is equivalent.

The information about each protocol supported in calls is to be found in the pages offered by the platform, as shown in the following screens:

HTTP POST specification:

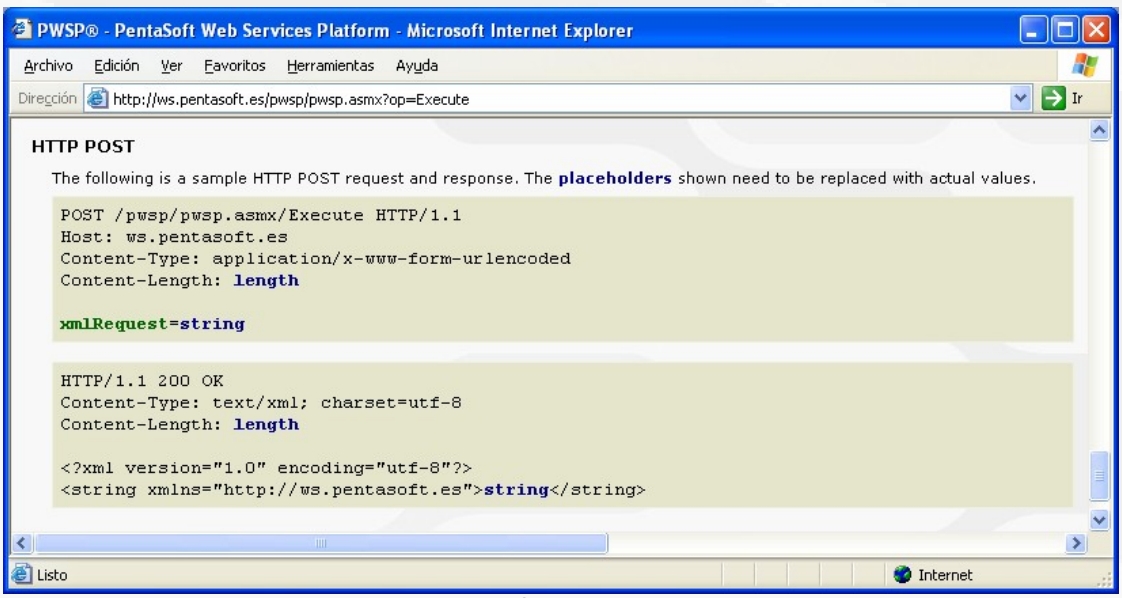


figure 13

®2007 PentaSoft Sistemas. All rights reserved.

HTTP GET specification:

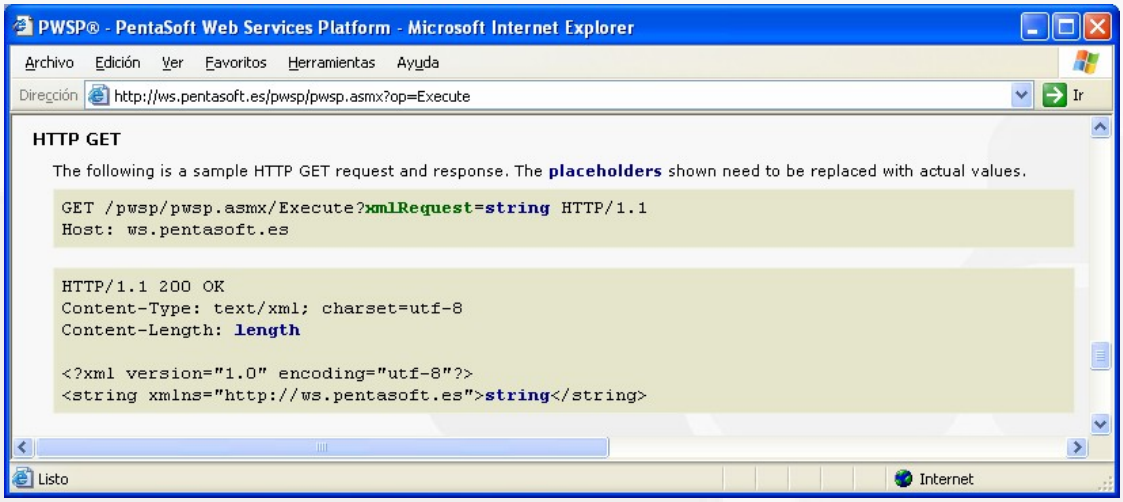


figure 14

SOAP 1.1 specification:

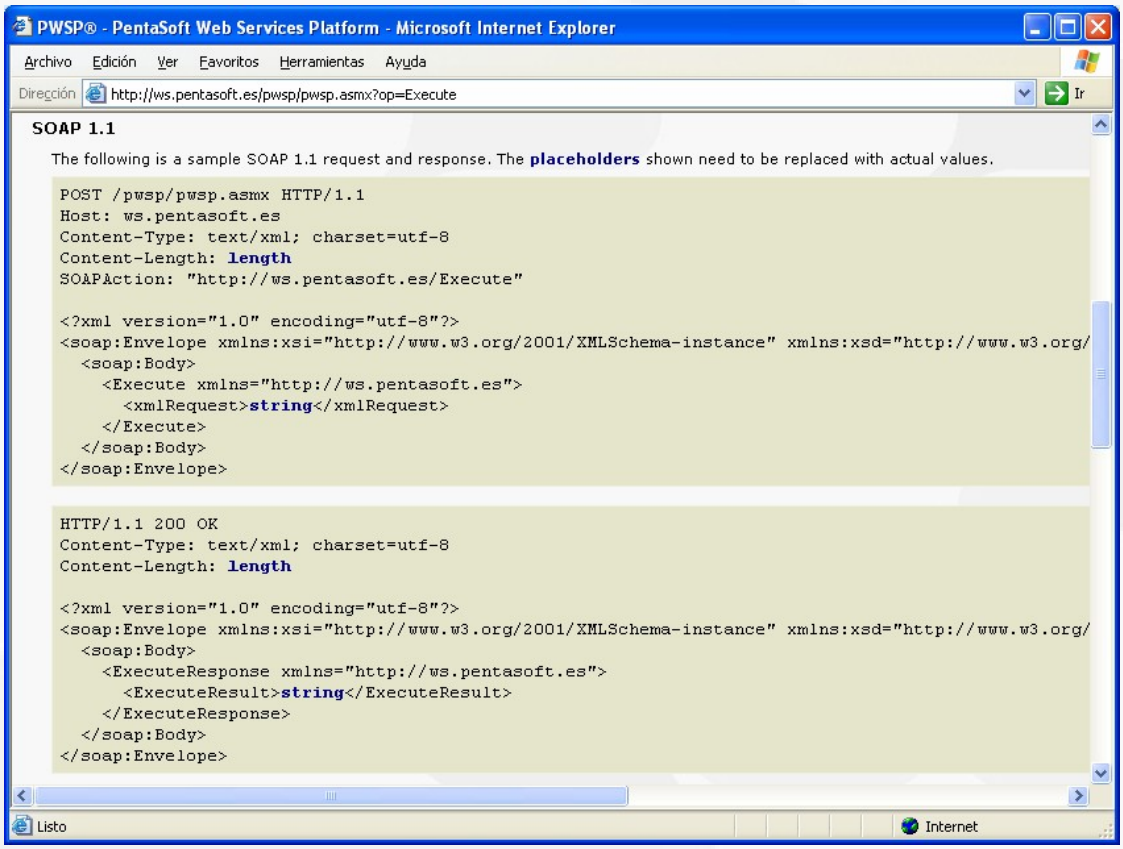


figure 15

®2007 PentaSoft Sistemas. All rights reserved.

SOAP 1.2 specification:

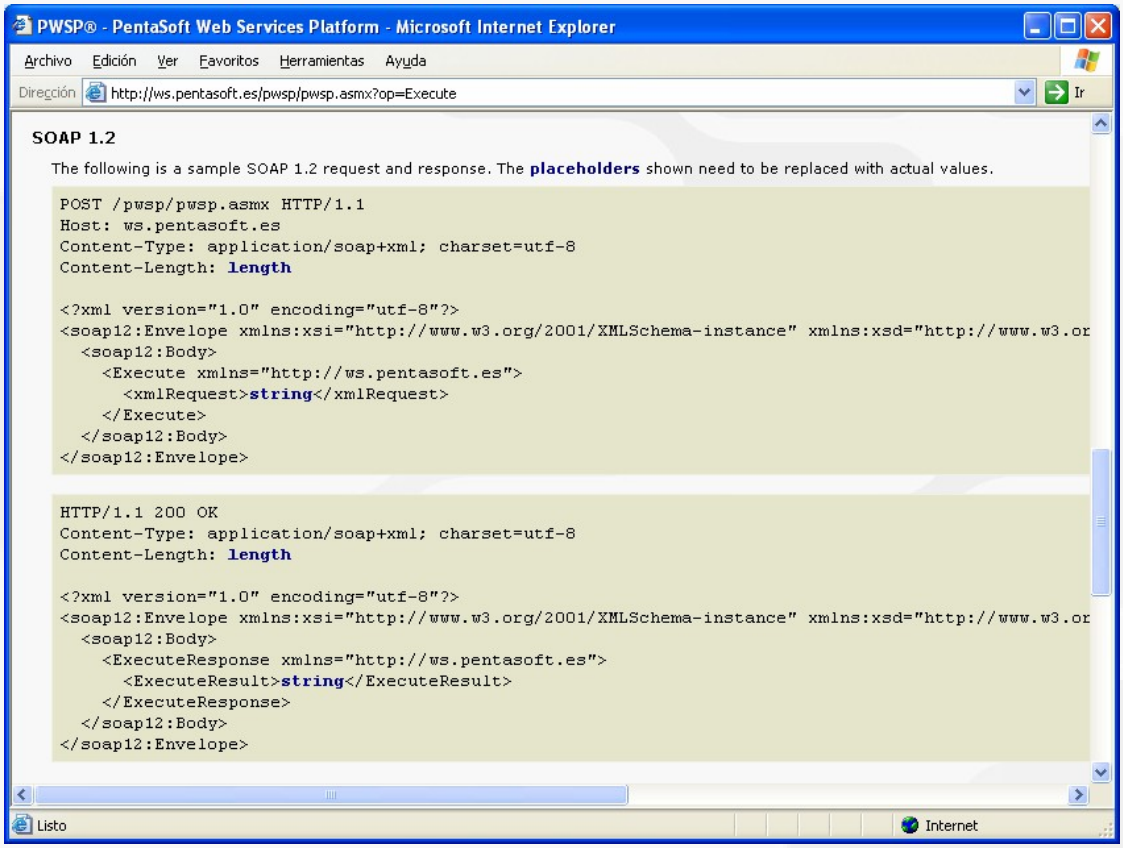


figure 16

®2007 PentaSoft Sistemas. All rights reserved.

4. Platform utilization examples

In the platform utilization process, it is advisable to follow the guideline of initially performing all the necessary tests on the page which the platform offers for this purpose (http://ws.pentasoft.es for production environments and http://ws.test.pentasoft.es for test environments), and once we have verified that we can correctly execute al the calls to the Web services which have been implemented, and that the information returned by these Web services is exactly that which is expected, we shall proceed to develop within our code the calls depending on the platform we are using.

We should remember that, irrespective of the technological environment in which our application or webpage functions and the programming languages in which it has been developed, the initial process of verifying the Web services on the platform test page is exactly the same.

It is advisable to carry out this process prior to starting development to include in our applications or webpages the calls to the platform, as this process saves development time and will considerably reduce implementation errors.

As we have mentioned in this document, the PWSP® platform can be invoked from any technological environment and any programming tool, and the only requirement is to have a connection to the Internet.

Due to the non-uniform nature of the systems which can connect to the platform, we have made two groups of examples, in view of the protocols used in executing the invocation of the web service, so as to cover all possible environments.

In the first group we can include the technological environments which are going to make the calls to the platform using the HTTP protocol, both GET and POST, and which could encompass all the pages or websites development using HTML and any script language or native programming language; it also includes any environment based on object models which are able to execute calls via HTTP either via querystrings or any similar method. It would also include in this category tools which function in web environments via the execution of a code based on flash type engines or similar and which can perform web invocations via HTTP.

In the second group we would include those web and desktop applications which can make calls via the SOAP protocol, both version 1.1 and 1.2. These environments include applications developed using Java, Microsoft.NET type tools and proprietary applications capable of making calls with Remoting utilities to invoke XML web services, basically via SOAP.

In all cases, this division does not attempt to include or group all the available environments or systems on the market, but to offer guidance on how to make the calls to the platform, because as we have said in this document, the platform can be used from virtually any system, irrespective of the technological environment or programming language, with the sole requirements of having some form of Internet connection.

The examples of utilization which we shall show implement a call to the platform with the invocation of the demo web service **pstsis06009** with the demo user data **Login: demologin Password: demopassword**. The information which will be entered are test data, as well as the results obtained from the platform, The examples can be used to perform all the necessary tests.

®2007 PentaSoft Sistemas. All rights reserved.

4.1. HTTP Querystring

Fro those environments in which we can make calls via HTTP using querystring, invoking the platform Web services is fairly simple, as the only information we need to supply is the value of the parameter which configures any call to the platform via this protocol.

Using as an example that given in the previous section, we shall see the appearance of the querystring which specifies the XML text stream to invoke the Logon method of the platform:

```
http://ws.pentasoft.es/pwsp/pwsp.asmx/Logon?xmlLogon=<Logon><Culture>es_ES</Culture><Login>demologin</Login>
<Password>demopassword</Password></Logon>
```

In blue is the address of the web service to access the platform, in fuchsia the name of the method we invoke in the service, in green the name of the parameter and from there to the end the serialised XML document of the Logon element. Within this parameter, in red, the values which are specified by the customer application in the call, which in this case are: the language in which we wish the platform to respond, the user and access password.

As a result of the call, the platform will return another serialised XML document as a stream of characters, in accordance with the schema (XSD file) which will contain the information of a Response element, and within this the security ticket specified in the Data element.

```
<?xml version="1.0" encoding="utf-8" ?><string
xmlns="http://ws.pentasoft.es"><Response><WebServiceId>pwsp</WebServiceId><MethodId>Logon</MethodId>
<Code>Ok</Code><Description>Acceso a la plataforma y ticket de seguridad generado
correctamente</Description><Message/><Data><![CDATA[SsA2YnehcDhmgbP/1K8kP/4/x4qguhVZfhnOJE8s+PEI]]></Data></
Response></string>
```

We have marked in orange and underlined the value of the security ticket which the platform has given us after having supplied the correct parameters. It is important to note that the security ticket is **solely** the value shown in orange and underlined and does not include any additional delimiter.

This information is important, because it is the value of the ticket which we must enter in subsequent calls to the execution method of the platform's web service so as to perform the invocation correctly. Otherwise, the platform will return an error code informing us that the security ticket supplied is not valid.

It is also important to note that **the value of the security ticket changes in each call to the Logon method** so the values we see in this example will not coincide with those we see in the tests we perform.

The security ticket contains information about the customer making the call, the system from which it is made, the time the ticket expires and other reserved information. For this reason, once we have obtained the ticket, we can use it to make calls from authorised systems until it expires.

Once the ticket has expired, the system answers with an error informing us of this situation. We only need to request a new ticket and continue with the rest of the process in the normal way. All these security methods guarantee the privacy and authenticity of the information exchanged between the systems, together with the non-intrusion of third parties in the process.

Once we have accessed the platform and obtained our security ticket, we can proceed to insert the customer data in our demo database. This operation is performed using the platform Execute method, so the querystring we need to specify has appears as follows:

®2007 PentaSoft Sistemas. All rights reserved.

```
http://ws.pentasoft.es/pwsp/pwsp.asmx/Execute?xmlRequest=<Request><Ticket>SsA2YnehcDhmgbP/1K8kP/4/x4qguhVZfhnOJE8s+PEI</Ticket><Culture>es_ES</Culture><WebServiceId>pstsis06009</WebServiceId><MethodId>AddCustomer</MethodId><Data><![CDATA[<Customer><Name>Evaristo</Name><Surname>Frutos</Surname><Age>45</Age></Customer>]]></Data></Request>
```

As a result of the call, the platform will return another XML document serialised as a stream of characters in accordance with the schema (XSD file) which will contain a Response element, which in turn will include the information on the return code. The result would be as follows:

```
<?xml version="1.0" encoding="utf-8" ?><string xmlns="http://ws.pentasoft.es"><Response><WebServiceId>pstsis06009</WebServiceId><MethodId>AddCustomer</MethodId><Code>Ok</Code><Description>El customer [Evaristo Frutos] ha sido añadido correctamente</Description><Message/><Data/></Response></string>
```

It is important to note that the correct return of the call to the platform is always a code with the value **Ok** and that we will always find it within the Response Code element. The first verification our customer code must make is that we receive this correct execution value from the platform.

If the execution is not correct, within the Code element we would receive an error code of those specified in the ResponseSet of the web service which in this case is pstsis06009.

If we wish to know all the possible return codes of a web service (in this case the demo web service pstsis06009), we must make the following calls:

- **Web service codes: pstsis06009 GetWebServiceResponseSet method. (Required)**
- **Platform: pwsp** codes **GetPlatformResponseSet method (Optional)**
- **Framework codes: pwsp GetFrameworkResponseSet method. (Optional)**

As shows in figure 1, section 2.1 of the technical features, the platform is organised on three levels, which comprise a library called Framework, the management layer of the Web services called Platform and the implementation of the Web services which in this case is the demo web service pstsis06009.

Each of the layers which build up the architecture can generate error messages. For this reason, in order to obtain all the possible error codes which the full architecture can offer, a method is provided for each level, so as to obtain the codes and return texts of each one. A customer application should be able to process at least the error codes of the web service; processing the platform and framework codes is optional.

Let us turn to the full example for obtaining all the possible return codes of the demo web service pstsis06009 as well as the underlying platform and framework.

Let us begin by the call we need to make to obtain the messages of the web service:

```
http://ws.pentasoft.es/pwsp/pwsp.asmx/Execute?xmlRequest=<Request><Ticket>SsA2YnehcDhmgbP/1K8kP/4/x4qguhVZfhnOJE8s+PEI</Ticket><Culture>es_ES</Culture><WebServiceId>pstsis06009</WebServiceId><MethodId>GetWebServiceResponseSet</MethodId><Data></Data></Request>
```

And the response of the platform:

```
<?xml version="1.0" encoding="utf-8" ?><string xmlns="http://ws.pentasoft.es"><Response><WebServiceId>pstsis06009</WebServiceId><MethodId>GetWebServiceResp
```


®2007 PentaSoft Sistemas. All rights reserved.

onseSet</MethodId><Code>Ok</Code><Description>El objeto ResponseSet implementado en el web service [pstsis06009] se ha leído correctamente</Description><Message /><Data><![CDATA[<ResponseSet><Version>App_Code.5s-azd71, Version=1.0.2447.22606, Culture=neutral, PublicKeyToken=null</Version><ResponseItems><ResponseItem><Key>CustomerDataInvalid</Key><Value>La información de customer suministrada es incorrecta</Value></ResponseItem><ResponseItem><Key>CustomerAddedSuccessfully</Key><Value>El customer [{0}] ha sido añadido correctamente</Value></ResponseItem></ResponseItems></ResponseSet>]]></Data></Response></string>

As we can see, all the codes which this web service can return are specified in the Key element of each of the ResponseItem within the web service ResponseSet, the texts associated with each code are specified in the Value element.

The customer system must be prepared to process any of these codes and act accordingly as regards the user, offering the information considered appropriate.

The specific messages of the platform can be obtained using the following call:

http://ws.pentasoft.es/pwsp/pwsp.asmx/Execute?xmlRequest=<Request><Ticket>SsA2YnehcDhmgbP/1K8kP/4/x4qquhVZfhnOJE8s+PEI</Ticket><Culture>es_ES</Culture><WebServiceId>pwsp</WebServiceId><MethodId>GetPlatformResponseSet</MethodId><Data></Data></Request>

And the response of the platform:

<?xml version="1.0" encoding="utf-8" ?><string xmlns="http://ws.pentasoft.es"><Response><WebServiceId>pwsp</WebServiceId><MethodId>GetPlatformResponseSet</MethodId> <Code>Ok</Code> <Description>El objeto ResponseSet implementado en el web service [pwsp] se ha leído correctamente</Description> <Message /> <Data><![CDATA[<ResponseSet> <Version>App_Code._rpnynpb, Version=1.0.2447.22179, Culture=neutral, PublicKeyToken=null</Version> <ResponseItems> <ResponseItem> <Key>TicketHasExpired</Key> <Value>El ticket de seguridad ha cadudado</Value> </ResponseItem> <ResponseItem> <Key>LogonUserTicketReturnException</Key> <Value>Se ha producido un error al intentar retornar el ticket de seguridad</Value> </ResponseItem> <ResponseItem> <Key>TicketUserIpDisabled</Key> <Value>La dirección ip especificada en el ticket de seguridad está deshabilitada</Value> </ResponseItem> <ResponseItem> <Key>PlatformMethodExecutionException</Key> <Value>Se ha producido un error al intentar ejecutar el método</Value> </ResponseItem> <ResponseItem> <Key>PlatformResponseSetReadSuccessfully</Key> <Value>El objeto ResponseSet implementado en la plataforma se ha leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>LogonUserTicketBuildException</Key> <Value>Se ha producido un error al intentar construir el ticket de seguridad</Value> </ResponseItem> <ResponseItem> <Key>ConfigFileInvalid</Key> <Value>El fichero de configuración es incorrecto</Value> </ResponseItem> <ResponseItem> <Key>LogonUserNotFound</Key> <Value>El usuario o la password especificados no existen</Value> </ResponseItem> <ResponseItem> <Key>PlatformMethodNotFound</Key> <Value>El método [{0}] no se encuentra implementado en la plataforma [{1}]</Value> </ResponseItem> <ResponseItem> <Key>LogonUserDisabled</Key> <Value>El usuario especificado ha sido deshabilitado</Value> </ResponseItem> <ResponseItem> <Key>PlatformWebServiceExecutionException</Key> <Value>Se ha producido un error al intentar ejecutar el web service</Value> </ResponseItem> <ResponseItem> <Key>WebServiceMethodsReadSuccessfully</Key> <Value>Los métodos públicos del web service se han leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>PlatformWebServiceDisabled</Key> <Value>El web service [{0}] está deshabilitado</Value> </ResponseItem> <ResponseItem> <Key>PlatformResponseSetReadException</Key> <Value>Se ha producido un error al intentar leer el objeto ResponseSet de la plataforma</Value> </ResponseItem> <ResponseItem> <Key>WebServiceSchemaDefinitionReadSuccessfully</Key> <Value>El fichero de definición del esquema [{0}] se ha leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>WebServiceMethodsReadException</Key> <Value>Se ha producido un error al intentar leer los métodos públicos del web service</Value> </ResponseItem> <ResponseItem> <Key>LogonUserOrPasswordInvalid</Key> <Value>El usuario o la password especificados son incorrectos</Value> </ResponseItem> <ResponseItem> <Key>TicketUserNotFound</Key> <Value>El usuario especificado en el ticket de seguridad no existe</Value> </ResponseItem> <ResponseItem> <Key>LogonUserDataReadException</Key> <Value>Se ha producido un error al intentar leer la información del customer que accede a la plataforma</Value> </ResponseItem> <ResponseItem> <Key>RequestFormatInvalid</Key> <Value>El formato del mensaje de solicitud es incorrecto</Value> </ResponseItem> <ResponseItem> <Key>TicketUserDisabled</Key> <Value>El usuario especificado en el ticket de seguridad está

®2007 PentaSoft Sistemas. All rights reserved.

deshabilitado</Value> </ResponseItem> <ResponseItem> <Key>PlatformWebServiceNotFound</Key> <Value>El web service [{0}] no se ha encontrado</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceSchemaDefinitionReadException</Key> <Value>Se ha producido un error al intentar leer el fichero de definición del esquema [{0}]</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceXmlDocumentationReadSuccessfully</Key> <Value>El fichero de documentación xml [{0}] se ha leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>WebServiceResponseSetReadSuccessfully</Key>
<Value>El objeto ResponseSet implementado en el web service [{0}] se ha leído correctamente</Value>
</ResponseItem> <ResponseItem> <Key>EnvironFileInvalid</Key> <Value>El fichero de definición del entorno es incorrecto</Value> </ResponseItem> <ResponseItem> <Key>LogonFormatInvalid</Key> <Value>El formato del mensaje de acceso a la plataforma es incorrecto</Value> </ResponseItem> <ResponseItem>
<Key>TicketFormatInvalid</Key> <Value>El formato del ticket de seguridad es incorrecto</Value>
</ResponseItem> <ResponseItem> <Key>TicketUserIpNotFound</Key> <Value>La dirección ip especificada en el ticket de seguridad no existe</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceResponseSetReadException</Key> <Value>Se ha producido un error al intentar leer el objeto ResponseSet del web service [{0}]</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceXmlDocumentationReadException</Key> <Value>Se ha producido un error al intentar leer el fichero de documentación xml [{0}]</Value> </ResponseItem> <ResponseItem> <Key>FromNameResourceKey</Key>
<Value>PWSP (Plataforma de Servicios Web de PentaSoft)</Value> </ResponseItem> <ResponseItem>
<Key>LogonUserSuccessfully</Key> <Value>Acceso a la plataforma y ticket de seguridad generado correctamente</Value> </ResponseItem> </ResponseItems> </ResponseSet>]]></Data> </Response></string>

Likewise, the specific messages of the framework are obtained using the following call:

http://ws.pentasoft.es/pwsp/pwsp.asmx/Execute?xmlRequest=<Request><Ticket>SsA2YnehcDhmgbP/1K8kP/4/x4qguhVZfhnOJE8s+PEI</Ticket><Culture>es_ES</Culture><WebServiceId>pwsp</WebServiceId><MethodId>GetFrameworkResponseSet</MethodId><Data></Data></Request>

And the response of the platform:

<?xml version="1.0" encoding="utf-8" ?><string xmlns="http://ws.pentasoft.es"><Response><WebServiceId>pwsp</WebServiceId><MethodId>GetFrameworkResponseSet</MethodId> <Code>Ok</Code> <Description>El objeto ResponseSet implementado en la plataforma se ha leído correctamente</Description> <Message /> <Data><![CDATA[<ResponseSet> <Version>Pentasoft, Version=1.0.0.4, Culture=neutral, PublicKeyToken=b6dc5df5bf20d98f</Version> <ResponseItems> <ResponseItem>
<Key>MethodNotFoundInWebService</Key> <Value>El método invocado [{0}] no se encuentra implementado en el web service [{1}]</Value> </ResponseItem> <ResponseItem> <Key>RequestObjectInvalid</Key> <Value>El objeto suministrado para realizar la solicitud no es válido</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceMethodsReadSuccessfully</Key> <Value>Los métodos públicos del web service se han leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>NumberMustBeGreaterThanZero</Key> <Value>El número debe ser mayor de cero</Value> </ResponseItem> <ResponseItem> <Key>ItemNotFoundInCollection</Key> <Value>El elemento [{0}] no se ha encontrado en la colección</Value> </ResponseItem> <ResponseItem> <Key>Ok</Key>
<Value>Correcto</Value> </ResponseItem> <ResponseItem> <Key>WebServiceSchemaDefinitionReadSuccessfully</Key>
<Value>El fichero de definición del esquema [{0}] se ha leído correctamente</Value> </ResponseItem>
<ResponseItem> <Key>WebServiceMethodsReadException</Key> <Value>Se ha producido un error al intentar leer los métodos públicos del web service</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceSchemaDefinitionReadException</Key> <Value>Se ha producido un error al intentar leer el fichero de definición del esquema [{0}]</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceXmlDocumentationReadSuccessfully</Key> <Value>El fichero de documentación xml [{0}] se ha leído correctamente</Value> </ResponseItem> <ResponseItem> <Key>WebServiceResponseSetReadSuccessfully</Key>
<Value>El objeto ResponseSet implementado en el web service [{0}] se ha leído correctamente</Value>
</ResponseItem> <ResponseItem> <Key>RequestSourceInvalid</Key> <Value>La solicitud solo puede provenir del web service de la plataforma</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceResponseSetReadException</Key> <Value>Se ha producido un error al intentar leer el objeto ResponseSet del web service [{0}]</Value> </ResponseItem> <ResponseItem>
<Key>WebServiceXmlDocumentationReadException</Key> <Value>Se ha producido un error al intentar leer el fichero de documentación xml [{0}]</Value> </ResponseItem> <ResponseItem> <Key>StringCantBeEmpty</Key>
<Value>La cadena de texto no puede estar vacía</Value> </ResponseItem> </ResponseItems>
</ResponseSet>]]></Data> </Response></string>

®2007 PentaSoft Sistemas. All rights reserved.

Now we could continue making all the calls we need to the Execute method of the platform, where the only thing we need is to vary the information contained in the various elements of the call request.

When the security ticket expires, we receive a response with an error code informing us of this situation. All we need to do is to invoke the Logon method again to obtain a new security ticket to be able to continue invoking web service methods.

®2007 PentaSoft Sistemas. All rights reserved.

4.2. Microsoft.NET Platform

Pending specifications.

®2007 PentaSoft Sistemas. All rights reserved.

5. Standards and rules specification

PentaSoft has a Project Development Methodology, PDM, which covers the tasks and activities defined by Metric 3 for the development of Object-oriented systems. During the lifecycle of the system development, the guidelines set out by this PDM methodology must be observed.

As regards the naming and coding of programs which form the PWSP® (Pentasoft Web Services Platform), the guidelines contained in the Document "Standards and Style Rules for program coding" must be followed.

PentaSoft also has an ISO 9001:2000 Quality System implemented, which establishes several control points to be carried out during each phase, as well as the management of incidents detected, thus ensuring that the product delivered complies with the quality levels required by the system.

®2007 PentaSoft Sistemas. All rights reserved.